VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

DETEKCE BITTORRENT PROVOZU V SÍTI LAN

MONITORING OF BITTORRENT TRAFFIC IN LAN

SEMESTRÁLNÍ PROJEKT

AUTOR **Bc. ONDŘEJ MIKULA**

BRNO 2023

Obsah

Kapitola 1

Úvod

Cílem tohoto projektu je vygenerovat a zachytit komunikaci protokolem BitTorrent v lokální síti (LAN), provést analýzu komunikace a na jejím základě navrhnout, implementovat a vyhodnotit možnosti pro detekci této komunikace v síťovém provozu. V úvodní části tohoto průvodního dokumentu je popsána architektura distribuované sítě a jsou popsány používané protokoly. Následně je uveden postup experimentu, jeho analýza a z ní vyvozené závěry. Dále je popsán vytvořený nástroj a metody pro detekci, které implementuje. V poslední kapitole tohoto dokumentu jsou vyhodnoceny implementované metody z hlediska obecnosti a možností, jak detekci zabránit.

Kapitola 2

Protokoly

Pro komunikaci v síti BitTorrent existuje více protokolů, které se vzájemně doplňují. Z hlediska této práce jsou nejpodstatnější protokoly BitTorrent, BitTorrent-DHT a uTP. BitTorrent je standardizován takzvanými BEPs (BitTorrent Enhancement Proposals)[1], jejichž funkce a formát lze přirovnat k RFC (Request For Comments).

V následujícím textu jsou používány termíny *peer* a *node*, které, ačkoli se zdají být zaměnitelné, není tomu tak. Peer je používán v kontextu komunikace protokolem BitTorrent a označuje se jím účastník v této síti. Naproti tomu node je zařízení[2] obecně v TCP/IP síti, které poskytuje informace pro vyhledávání zdrojů v síti BitTorrent. Typicky má jedno zařízení obě role a BitTorrent klient implementuje obojí, obecně může mít jen jednu z nich.

Připojit se do BitTorrent sítě lze dvěma způsoby, buďto pomocí DHT protokolu (viz dále) nebo tzv. trackeru. Tracker je server (jako ve standardní klient/server architektuře), který má informace o torrentech, zúčastněných peers a předává je nově se připojujícím peers. Komunikace s trackerem probíhá přes HTTP nad TCP, nebo „UDP Tracker Protocol for BitTorrent“[3], který snižuje režii přenosu využitím UDP místo TCP.

2.1 Protokol BitTorrent

Protokol je specifikován v „The BitTorrent Protocol Specification,“ BEP 3[4]. Na transportní vrstvě používá TCP nebo uTP. Pro adresování peers využívá 160 bit hash funkcí SHA–1. Komunikace potokolem BitTorrent začína provedením handshake, který začíná 1 byte, který pro kompatibilitu s jinými protokoly udává délku následujícícho řetězce „BitTorrent protocol“. Následují rezervované byty pro rozšíření, SHA–1 hash požadovaného zdroje a peer ID. Dotazovaný peer odpoví odpovídající zprávou Handshake s přidanou zprávou Extended a klient odpoví svou zprávou Extended. Následující zprávy jsou jednoho z 9 typů, který je obsažen v 1. byte dané zprávy:

- 0 - choke – nastavuje pozastavení komunikace, aby si klient nevyměňoval pieces se všemi peers, ale pouze se čtyřmi s nejlepším bitrate,
- 1 - unchoke – opak předchozího,

[1] Seznam všech BEP je zde: https://www.bittorrent.org/beps/bep_0000.html.

[2] Dále je také používáno „zařízení“ ve významu „client/server“, jakožto termín pro účastníka v distribuované síti, z důvodu zkrácení textu a čitelnosti.

[3] https://www.bittorrent.org/beps/bep_0015.html.

[4] https://www.bittorrent.org/beps/bep_0003.html.

- 2 - interested – uvádí, o které pieces má klient zájem,
- 3 - not interested – opak předchozího,
- 4 - have – oznamuje, že peer stáhl a ověřil piece s poslaným ID,
- 5 - bitfield – obsahuje bitové pole kódující, kterými pieces klient disponuje,
- 6 - request – požadavek piece s indexem a délkou,
- 7 - piece – obsahují data daného piece a jeho index,
- 8 - cancel – ruší dříve poslané zprávy request.

Stav komunikace mezi dvěma peers je dán hodnotami choke a interested. K přnosu pieces dochází, když jsou peers unchoked a interested zároveň.

2.2 Protokol BitTorrent-DHT

Protokol je specifikován v „DHT Protocol," BEP 5[5]. Poskytuje způsob vyhledávání obsahu v síti bez využití trackerů. Vychází z protokolu Kademlia a na transportní vrstvě používá UDP. Zkratka DHT znamená „distributed sloppy hash table". Nodes jsou adresovány tzv. node ID, které má 160 bitů, stejně jako infohash obsahu v síti. Každý node si udržuje směrovací tabulku (DHT), která obsahuje tzv. sousedy.

Adresní prostor DHT se dělí na takzvané buckets, které jsou relativní k danému node. Každý bucket může obsahovat 0 až K nodes. Pokud je bucket plný a node, který tabulku vytváří, sám patří do daného bucket, může být rozštěpen na dva menší buckets poloviční velikosti. Tím je zajištěno, že každý node si může udržovat více sousedů ve své blízkosti, zatímco v vzdálenějším prostoru adres jen omezený počet. Nodes v DHT se rozlišují na „good", „questionable" a s „unknown status", podle toho, zda udržují sousednost. Dotazy a odpovědi v DHT protokolu jsou formátovány dle KRPC protokolu.

2.2.1 KRPC protokol

Obsah KRPC zpráv je zakódován pomocí bencoding. Zprávy obsahují jeden slovník (dictionary), ve kterém jsou páry klíč–hodnota s klíči t, y a v. Hodnota klíče t, transaction, slouží ke spárování dotazu a odpovědi. Hodnota klíče y určuje typ zprávy a může nabývat hodnot q (query, dotaz), r (response, odpověď) a e (error, chyba). Hodnota klíče v udává verzi DHT klienta a není povinná.

KRPC specifikuje 3 typy zpráv: query, response a error. Slovník zprávy typu query obsahuje navíc klíč q, který specifikuje dotaz a klíč a s argumenty k dotazu. Slovník pro zprávu response má navíc klíč r obsahující odpovědi. Zprávy typu error nejsou z hlediska detekce BitTorrent komunikace podstatné. Dotaz v query může být:

- *ping* pro udržování sousednosti,
- *find_node* pro hledané node ID odpoví buďto informacemi o node, nebo 8 nejbližšími node ID v tabulce dotazovaného (dotazování pak probíhá iterativně, dokud nedorazí dotaz k node, který hledaný node zná),

[5] https://www.bittorrent.org/beps/bep_0005.html.

- *get_peers* provede dotaz na infohash. Pokud má dotazovaný uložené informace o peers daného torrentu, pošle je v odpovědi, jinak odpoví K[6] nejbližšími nodes ze své DHT (opět probíhá hledání iterativně),
- *announce_peer* oznámí, že peer svázaný s oznamujícím node se zapojil do torrentu, aby si odpovědný node mohl tuto informaci uložit a předávat dalším nodes při příštích dotazech.

2.3 Protokol uTP

Pro specifické potřeby komunikace BitTorrent vznikl protokol uTorrent Transport Protocol (dále uTP), který může na transportní vrstvě nahradit jinak používaný protokol TCP. uTP je standardizován dokumentem BEP 29[7]. uTP je vnořen do protokolu UDP. Motivací pro využití tohoto protokolu je vhodnější využití šířky pásma připojení. BitTorrent často navazuje více TCP spojení současně (k více peers), což vede ke znevýhodnění jiných spojení, jelikož se operační systém snaží obsloužit všechna rovnoměrně. Jelikož přijímání a odesílání BitTorrent dat má pro uživatele typicky nízkou prioritu a často probíhá kontinuálně na pozadí, je cílem uTP co nejvíce využít šířku pásma, ale zároveň neomezovat jiný síťový provoz.

uTP poskytuje spolehlivost a doručení v pořadí. Implementuje vlastní mechanismus pro řízení zahlcení.

Vzhledem k tomu, že uTP cílí transparentně nahradit protokol TCP a jeho využití není povinné (a z toho důvodu je detekce BitTorrent pomocí uTP nedostatečná), není jeho rozbor pro potřeby tohoto projektu více nutný.

[6] Hodnota není blíže specifikována.
[7] https://www.bittorrent.org/beps/bep_0029.html.

Kapitola 3

Provedené experimenty s klientem

Pro seznámení s protokolem BitTorrent a dalšími používanými protokoly a získání vzorku dat pro následně implementovaný program byly provedeny dále popsané experimenty. Zachycená data z experimentů jsou uložena ve složce `data` ve formátu PCAPNG a CSV. Experimenty byly prováděny především s klientem qBittorrent a metody pro detekci jsou navrženy pro tohoto klienta, pro porovnání a zjištění obecnosti byly jednoduché experimenty zahrnující inicializaci a stažení souborů provedeny také s dalšími klienty.

3.1 Prostředí

Experiment byl proveden ve virtualizovaném systému Manjaro Linux verze 22.0.2 s BitTorrent klientem qBittorrent[1] verze 4.5.2 a síťový provoz byl monitorován nástrojem Wireshark. Systém byl spuštěn v tzv. Live CD modu, tedy bez persistence, po spuštění byla provedena synchronizace s repozitáři, nainstalovány balíčky `qbittorrent`, `transmission-qt`, `ktorrent`, `bitstormlite`, `vuze`, `deluge`[2], a `wireshark` a proveden experiment. Systém byl propojen s hostujícím operačním systémem (Manjaro Linux 22.1.0) pouze jedním virtuálním síťovým rozhraním, na kterém byl na straně virtualizovaného systému zachytáván provoz. IP adresa klienta byla 10.0.2.15. Všechny testované programy byly provozovány v základním, nezměněném nastavení.

3.2 Průběh

Bylo provedeno několik experimentů, zahrnujících různé části životního cyklu klienta, jako inicializace (1. spuštění), získávání sousedů, vyhledání a stažení souboru a nakonec odhlášení při vypnutí klienta. Názvy souborů se zachycenými daty reflektují obsažené fáze a mají příponu dle klienta, se kterým byl daný experiment proveden, například *qbit-* pro qBittorrent.

Následující analýza využívá především data ze souboru `qbit-init-download.pcapng`. V něm jako první proběhla inicializace, která se typicky provádí jen při prvním spuštění. Následně proběhla fáze „Získávání sousedů“, kdy klient kontaktoval další peers. Poté byl, na základě vstupu od uživatele, vyhledán a stažen z BitTorrent sítě požadovaný soubor[3].

[1] https://www.qbittorrent.org/. qBittorrent je grafický frontend, pro komunikaci používá knihovnu libtorrent.

[2] Každý klient byl nainstalován samostatně na nově spuštěný Live CD systém, například kvůli DNS cache.

[3] Konkrétně z torrentu dostupného na adrese https://academictorrents.com/details/632090223863ebe433f0e598b4819638e4d00470.

Nakonec došlo k úplnému vypnutí programu klienta (ne pouze minimalizaci do lišty, což je standardní chování qBittorrent).

3.2.1 Inicializace

V souboru se zachycenými daty začíná komunikace qBittorrent klienta 37. paketem přibližně 5.2 sekund po začátku zachytávání. Klient jako první provede dotazy na DNS server (v tomto případě 147.229.190.143, nakonfigurovaný v systému). Dotazuje se na A (IPv4) i AAAA (IPv6) záznamy na tyto hostname v uvedeném pořadí:

- dht.libtorrent.org
- router.bittorrent.com
- router.utorrent.com
- dht.transmissionbt.com
- dht.aelitis.com

V průběhu také provede DNS dotaz na „download.db-ip.com“, a na přeloženou IP adresu 104.26.5.15 se připojí k TCP portu 443 protokolem TLS a obdrží 3604 kB dat. Vzhledem ke jménu serveru se pravděpodobně jedná o stažení databáze, která mapuje IP adresy na geolokační údaje.

V 5. sekundě také klient pomocí IGMPv3 oznámí připojení k multicast skupinám 239.192.152.143 a 239.255.255.250.

Packetem 160, přibližně 0.4 sekundy po prvním relevantním datagramu, začíná BitTorrent-DHT komunikace. Klient posílá zprávy typu request s příkazem *get_peers* v uvedeném pořadí na tyto IP adresy:

- 34.229.89.117 (odpovídá dht.aelitis.com)
- 67.215.246.10 (odpovídá router.bittorrent.com)
- 82.221.103.244 (odpovídá router.utorrent.com)
- 87.98.162.88 (odpovídá dht.transmissionbt.com)
- 185.157.221.247 (odpovídá dht.libtorrent.org)

Zprávy jsou, až na *transaction_id* a dotazovaný *info_hash*, identické. Prvních 12 byte *info_hash* se ve všech dotazech shoduje s posílaným *id*, zbylých 8 byte se zdá být náhodných. Tento mechanismus může sloužit k nalezení sousedů, kteří jsou relativně blízko klienta (sdílejí 12 byte prefix). Není zřejmé, proč klient získává informace o sousedech pomocí dotazu *get_peers* místo *find_node*, ačkoli zjevně fungují oba přístupy.

V dotazech ve slovníku *a* se vyskytuje položka *bs*, ke které se sice autorovi nepodařilo nalézt dokumentaci, ale dle komentáře v implementaci libtorrent[4] označuje dotaz prováděný v rámci bootstrap, což odpovídá pozorované komunikaci.

Z 5 odeslaných dotazů klient obdržel 3 odpovědi (u zbylých pravděpodobně nebyl doručen dotaz nebo odpověď a vzhledem k použití UDP o tom nebyl klient informován), které

[4] https://github.com/arvidn/libtorrent/blob/95afba30e9acfd8411076a0b05f7ca69bf33b9bb/src/kademlia/refresh.cpp#L67

obsahovaly informace o 8, 3 a 16 nodes. V případě první obdržené odpovědi obsahovalo všech 8 záznamů informace o stejném node, ve zbylých dvou odpovědích byly informace o různých nodes.

3.2.2 Získávání sousedů

Ihned po obdržení první odpovědi od bootstrap node klient odeslal dotaz *get_peers* na nově zjištěný node. Ten v případě zachycené komunikace neodpověděl (nebo se, pravděpodobněji, zpráva ztratila), ale mezitím přišly odpovědi od zbylých bootstrap serverů. V dotazech na získané peers klient uváděl jiné ID než při komunikaci s bootstrap nodes.

Dotazy na zjišťování sousedů pokračovaly v experimentu `qbit-init-download.pcapng` až do ukončení programu klienta, na dotazovaném *info_hash* lze pozorovat, že klient postupně vyhledával nodes ve vzdálenějších místech adresního prostoru. Z dalších provedených experimentů vyplývá, že qBittorrent přestane vyhledávat další sousedy po dosažení přibližně 160 sousedů.

Zprávy typu ping byly zachyceny přibližně 4.7 minut po otevření klienta.

3.2.3 Vyhledání zdroje

V čase 150 sekund po začátku zachytávání proběhl import souboru `thermodynamics.docx-632090223863ebe433f0e598b4819638e4d00470.torrent` do klienta. Prvním relevantním paketem je paket 907, kterým byl proveden DNS dotaz na doménu „drive.google.com“, se kterou bylo následně navázáno TLS spojení. URL na této doméně je zmíněno v položce *url-list* v souboru torrentu. Následuje 5 dotazů typu *get_peers* s hash požadovaného souboru na peers s následujícími ID:

- 72d0b59b6a6223bcc4bc23b324f0040b98fa6b9d
- 7bb5ef6ad3c9a08f76a103f94d7b47cb25ea157c
- 47e05be39a1dc712f4e151f87a0b06bdfbc0c254
- 5eaf3dc789e4b77cd3e02923259d8c5ccf32b20a
- 5d8f7799b6f44d909b59ce9dca3d26417e26c257

Tato ID jsou k ID hledaného obsahu nejblíže ze sousedů klienta. Se všemi z nich již předtím klient navázal kontakt zprávou *get_peers*.

Následují DNS dotazy na „academictorrents.com“, „ipv6.academictorrents.com“, „tracker.opentrackr.org“ a „tracker.openbittorrent.com“, které se vyskytují v položce *announce*, respektive seznamu *announce-list* v torrentu.

3.2.4 Stažení souboru

První zpráva protokolem BitTorrent je v paketu 1013 a jedná se o handshake. Dotazovaný peer odpověděl zprávou Handshake Extended a následně klient zaslal zprávu Extended. Následně peer poslal zprávu typu bitfield s informacemi o pieces, které má dostupné, klient poslal zprávu interested a peer poslal zprávu unchoke, čímž byla komunikace uvedena do stavu, kdy mohla být přenášena data.

Klient následně posílal zprávy request na jednotlivé pieces a peer odpovídal požadovanými pieces. Úspěšné stažení a ověření integrity klient potvrzoval zprávou Have. V zachycené

komunikaci klient komunikuje pouze s jedním peer, který má všechny pieces a tím komunikace připomíná více architekturu klient / server. V experimentech s více torrenty a většími soubory probíhá přenos více distribuovaně.

3.2.5 Odhlášení

Při odhlášení klient odešle protokolem IGMPv3 a ICMPv6 informaci a odhlášení z multicast skupin 239.192.152.143 a 239.255.255.250.

3.3 Vyhodnocení

V zachycené komunikaci se podařilo identifikovat jednotlivé fáze provozu BitTorrent klienta a doménová jména, IP adresy a node ID, se kterými byla provedena komunikace. V komunikaci byly nalezeny také následující protokoly, které nebyly v implementační části pro detekci využity:

- LSD (Local Service Discovery), odeslaný na multicast adresu, obsahující header „BT-SEARCH * HTTP/1.1“ a infohash požadovaného torrentu,
- PCPv2 (Port Control Protocol) požadavek na mapování portu 8990 v NAT,
- NAT–PMP (NAT Port Mapping Protocol) s požadavkem na získání externí IP adresy (opcode 0) v experimentu s klientem Transmission,
- IGMPv3, ICMPv6, kterými se klient připojoval k multicastovým skupinám a při ukončení klienta od nich se odpojil.

Kapitola 4

Nástroj pro analýzu provozu BitTorrent

Vypracovaný nástroj `bt-monitor` se nachází ve složce `src` a je implementován v jazyce Python (verze 3). Základní přepínače programu odpovídají závazné specifikaci, přepínač rtable není implementován. Navíc implementuje tyto přepínače, které provádí detekci jinými způsoby:

- -dns
- -http
- -ports
- -ping
- -tcpudpport

U některých přepínačů program `bt-monitor` přijímá vstup ve formátu PCAP a u jiných ve formátu CSV. V případě nekompatibilní kombinace oznámí chybu uživateli. Ukázky spuštění programu jsou v přiloženém souboru Makefile, kde jsou také pravidla pro spuštění nad všemi soubory se zachycenými daty.

Ve stejné složce se také nachází shell skript `pcapng_to_csv.sh`, který vstupní soubor typu `pcapng` zpracuje do souboru CSV. Vstupní soubor je specifikován jako 1. poziční argument a výstupní soubor je pojmenován stejně jako vstupní, s přidanou příponou .csv. Skript zpracovává data pomocí programu `tshark`.

4.1 Detekce inicializace

Pro detekci inicializace program `bt-monitor` implementuje 3 různé metody.

- 1. metoda detekuje položku *bs* (*bootstrap*) v požadavku get_peers nebo find_nodes. Tato metoda je spolehlivá, ale funguje pouze pro klienty založené na knihovně libtorrent, která položku *bs* do požadavku přidává, ačkoliv není ve specifikaci BitTorrent vůbec uvedena.
- 2. metoda je obecnější a funguje na principu vytvoření seznamu všech nodes, které byly vráceny v odpovědi na dotaz get_peers nebo find_nodes. Při druhém průchodu

vypíše ty kontaktované nodes, které nejsou v seznamu z předešlého kroku, a tudíž se o nich klient dozvěděl jinak než dotazováním protokolem DHT.

- Ve 3. metodě jsou nejdříve shromážděny IP adresy z DNS odpovědí a následně jsou vypsány kontaktované nodes, jejichž IP adresy byly předtím získány DNS dotazem. Tato metoda je také více obecná, její nevýhodou je, že spoléhá na provedení DNS dotazu. V případě, že klient má bootstrap nodes uložené jako IP adresy, je nepoužitelná.

4.2 Detekce peers

Detekce peers[1] je prováděna parsováním odpovědí na dotazy *get_peers* a *find_node*. V případě, že je zachycena odpověď na jeden z těchto dotazů (čímž je zřejmé, že peer byl kontaktován a odpověděl), uloží se IP adresa, port a node ID odpovídajícího peer do tabulky detekovaných peers, která je nakonec vypsána. Vzhledem k výskytu klientů, kteří nekomunikují protokolem DHT (z testovaných KTorrent a BitStrom Lite) byla navíc implementována detekce BitTorrent zpráv typu piece (7), jejichž odesilatel je také považován za peer. Nevýhodou této metody je, že nelze získat node ID.

4.3 Detekce přenosu souborů

Tento switch vypíše všechny infohash požadovaných torrentů a pro chunks přenášených dat torrentu jejich index, velikost, počet přenosů a zdrojovou IP adresu a port.

Parsování infohash a dalších položek je prováděno na úrovni tsharku. Celková odhadovaná velikost je vypočtena jako součet všech unikátních chunks (tedy nejsou započítávány duplicitní přenosy chunks).

Čas stahování je vypočten jako rozdíl času zachycení posledního chunk a času dotazu na infohash. V případě, že dotaz na infohash není zachycen, je nahrazen časem prvního zachyceného chunk, takže program dokáže odhadhout čas i bez zachycení dotazu na infohash.

4.4 Další implementované techniky

Kromě přepínačů specifikovaných zadáním byly implementovány další, které umožňují detekovat BitTorrent provoz jinými metodami.

4.4.1 Klíčová slova v DNS

Tento způsob detekce provede prohledání DNS datagramů a vypíše všechny dotazované domény, v jejichž názvu je jedno z klíčových slov „torrent“, „tracker“ nebo „dht“.

4.4.2 Klíčová slova v HTTP

Analyzuje obsah UDP i TCP datagramů s cílovými nebo zdrojovými porty 80, 8080, 8000, 8081, 8888 a hledá jedno z klíčových slov (respektive řetězců) „/announce“ nebo „/scrape“.

[1] Je potřeba definovat termín „peer“ v tomto kontextu. V rámci tohoto přepínače je chápán jako účastník v síti, se kterým alespoň jednou proběhla **obousměrná** komunikace. Z toho vyplývá, že „peers“, kteří neodpověděli na dotaz, nebo „peers“ jejichž ID a adresu vrátí na dotaz node ale nejsou následně kontaktováni, za peers považováni nejsou.

4.4.3 Detekce pomocí známých portů

Vypíše zachycené IP adresy, porty a směr komunikace pro datagramy, jejichž zdrojový nebo cílový port je jeden z tzv. „well known ports“. Detekuje TCP i UDP. Detekovány jsou tyto porty:

- 6881–6889, specifikované v BEP 3[2],
- 8990, který byl zachycen v analyzované komunikaci,
- 6969, používaný pro komunikaci s trackery[3],

4.4.4 Zprávy ping

Detekuje zprávy typu ping protokolem DHT. Vypíše zdrojovou a cílovou IP adresu a port.

4.4.5 TCP a UDP provoz na stejném portu

Provádí detekci portů jednotlivých IP adres, na kterých byla zachycena UDP a zároveň TCP komunikace, což může indikovat BitTorrent provoz. Detekce je provedena jak pro zdrojové, tak cílové porty.

[2] https://www.bittorrent.org/beps/bep_0003.html.

[3] https://superuser.com/questions/1706564/which-ports-need-to-be-allowed-through-firewall-to-seed-torrent-using-transmissi.

Kapitola 5

Vyhodnocení

Vytvořený program dokáže detekovat BitTorrent komunikaci, zjistit jak IP, tak BitTorrent adresy komunikujících stran, získat infohash torrentu, který klient požadoval, přenesené chunks dat torrentu a jinými způsoby detekovat v zachyceném síťovém provozu komunikaci BitTorrent klienta. Implementované metody však mají omezení a existují způsoby, jak detekci předejít.

Jedním z nich je použití šifrování, které musí umět podporovat obě strany. Vzhledem k absenci protokolu BitTorrent a výskytu TLS v komunikaci klienta Transmission se autor domnívá, že zachycené stažení souboru proběhlo šifrovaně.

Dalším způsobem pro předcházení detekce může být šifrování DNS dotazů a odpovědí, využívání jiných než známých portů pro komunikaci a vyhnutí se používání jednoho portu pro TCP a UDP komunikaci.

Transmission implementuje obfuskaci DNS dotazů střídáním velkých a malých písmen, jako například „trAckEr.oPenTRaCKR.OrG“. Toto je ošetřeno převedením dotazu do malých písmen před další prací s ním a program dokáže detekovat i tyto DNS dotazy.

Klienti KTorrent a BitStrom Lite nepoužívají protokol DHT a není možné detekovat jejich peers jinak funkční metodou detekce DHT. To bylo vyřešeno přidáním detekce peers z BitTorrent zpráv typu piece.

Vzhledem k přílišné složitosti metoda download nepodporuje detekci více stahování v jednom souboru. V takovém souboru dokáže nalézt více infohashes, ale přiřadit jednotlivé pieces k torrentům a vypočítat délky by bylo značně náročné.

V případě detekce inicializace a ping zpráv je možné (a tato situace v experimentu nastala), že peer, na který je ping směřován, není dostupný. V takovém případě může být vrácena klientovi ICMP zpráva, například Host unreachable, která obsahuje původní ping zprávu. Při parsování nástrojem tshark se tak dostanou do sloupce se zdrojovou a cílovou adresou obě adresy zároveň. K výskytu této chyby dochází již v programu tshark a tato situace byla řešena jednoduchou detekcí znaku čárky v poli pro IP ardesu. Příklad této chyby lze nalézt například v packetu č. 1355 v souboru `qbit-initonly.pcapng`.

Tabulka 5.1 ukazuje, pro které klienty byly implementované metody schopny v zachycených datech detekovat BitTorrent komunikaci. Z tabulky vyplývá, že navržené metody peers a dns byly schopny detekovat BitTorrent komunikaci u všech vyzkoušených klientů, metody download a ports lze s jedním nedetekovaným klientem také považovat za úspěšné, zatímco efektivita ostatním metod je značně slabší. Autorovi se nepodařilo zjistit, proč je metoda http úspěšná pouze u klienta qBittorrent, možnými důvody jsou šifrování HTTP komunikace ostatními klienty nebo špatné argumenty pro parsování programem tshark. I přes úsilí tuto metodu zobecnit na TCP i UDP provoz a zahrnout další časté HTTP porty ne-

bylo dosaženo zvýšení úspěšnosti této metody. Pro neúspěšnost metody ping byly pořízeny další záznamy komunikace trvající alespoň 15 minut, přesto u některých kllientů nebyly zprávy ping detekovány. Podobnost výsledků pro qBittorrent a Deluge je pravděpodobně způsobena používáním stejného backendu, libtorrent.

Klient	**init I**	**init II**	**init III**	**peers**	**download**
qBittorrent	✓	✓	✓	✓	✓
Transmission	x	x	x	✓	x
BitStrom Lite	x	x	x	✓	✓
Deluge	✓	✓	✓	✓	✓
KTorrent	x	x	x	✓	✓
Vuze	x	✓	x	✓	✓
	dns	**http**	**ports**	**ping**	**tcp and udp**
qBittorrent	✓	✓	✓	✓	✓
Transmission	✓	x	✓	✓	x
BitStrom Lite	✓	x	x	x	x
Deluge	✓	x	✓	✓	✓
KTorrent	✓	x	✓	x	x
Vuze	✓	x	✓	x	✓

Tabulka 5.1: Schopnost navržených metod detekovat komunikaci vyzkoušených klientů.

Kapitola 6

Závěr

V rámci tohoto projektu byla vygenerována a zachycena komunikace šesti BitTorrent klientů se sítí, provedena manuální analýza komunikace klienta qBittorrent (a částečně také Transmission) a na jejím základě vytvořen nástroj, který implementuje několik metod pro detekci BitTorrent komunikace. Nakonec byly implementované metody otestovány a diskutovány jejich limity a úspěšnost.